영상처리 Python 실습 과제 2

201620350 김지영

Histogram equalization을 위해서 다음과 같은 python program을 작성하고 실행하여 결과를 관찰하시오.

1. 흑백 영상을 histogram equalization하기 위한 python program을 작성하여 lena_512_low1.png, lena_512_low2, boats_512_high.png, lena.png에 대해서 실행하고 contrast stretching을 적용하였을 때 와 차이를 논하시오. (function을 사용하지 않고, histogram equalization부분을 중첩된 for문을 사용하여 직접 작성할 것)

(1) 그림1 - lena_512_low1.png

histogram equalization

[정답] hist_eq.py

```
import numpy as np
from matplotlib import pyplot as plt

# original image & histogram
inimg = plt.imread('boats_512_high.png')
inimg = np.uint8(255*inimg)

plt.imshow(inimg,'gray',vmin=0,vmax=255)
plt.show()

plt.hist(inimg.flatten(),256,[0,256])
plt.show()

# histogram equalization
h,w = inimg.shape

freq = np.zeros(256)
prob = np.zeros(256)
```

```
cprob = np.zeros(256)
eqval = np.zeros(256)
outimg = np.zeros(inimg.shape)

for i in range(h):
    for j in range(w):
        pval = inimg[i,j]
        freq[pval] = freq[pval]+1
for i in range(256):
    prob[i] = float(freq[i]) / float(h * w)
    if(i>0):
        cprob[i] = cprob[i-1] + prob[i]
    else:
        cprob[i] = prob[i]
    eqval[i] = round(255 * cprob[i])
for i in range(h):
    for j in range(w):
        outimg[i,j] = eqval[inimg[i,j]]

# equalized image & histogram
plt.imshow(outimg,'gray',vmin=0,vmax=255)
plt.show()

plt.hist(outimg.flatten(),256,[0,256])
plt.show()
```

과제제출 코드

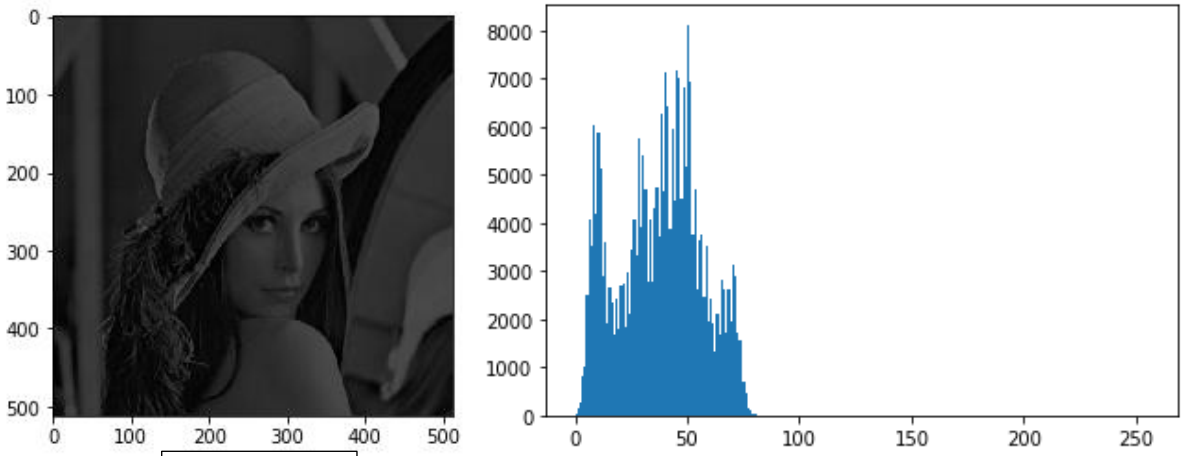
- Python Code

```
import scipy
import numpy as np
from scipy import ndimage
from matplotlib import pyplot as plt
```

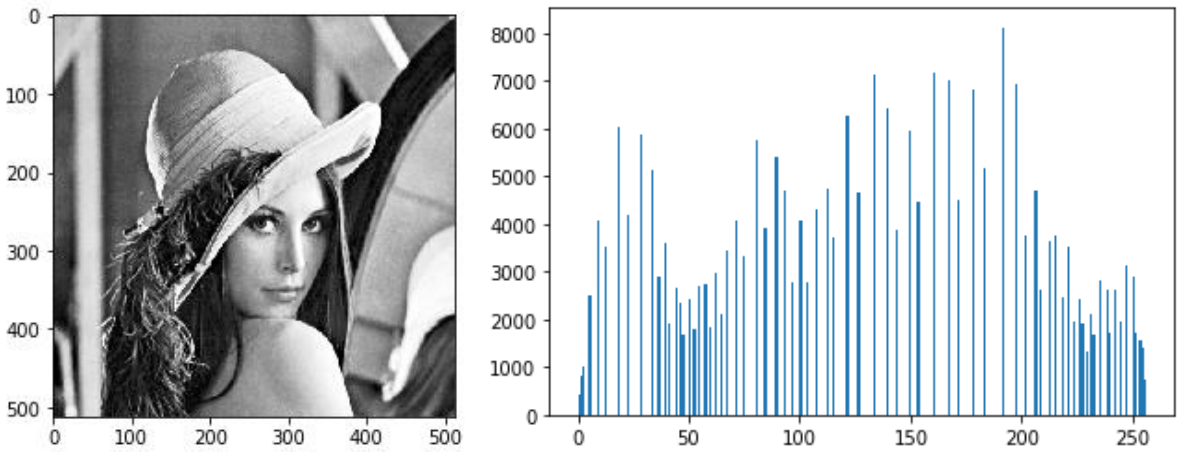
```
inimg = plt.imread('lena_512_low1.png')
inimg = np.uint8(255*inimg)

plt.imshow(inimg,'gray',vmin=0,vmax=255)
plt.show()
plt.hist(inimg.flatten(),256,[0,256])
plt.show()

count = np.zeros(256).astype('uint32')
cdf = np.zeros(256).astype('uint64')
move = np.zeros(256)

while 1:

    # Image histogram 생성
    h,w = inimg.shape
    for i in range(h):
       for j in range(w):
            x = inimg[i][j]
            count[x] += 1

    # 누적 histogram 생성
    for i in range(len(count)):
        cdf[i] = cdf[i-1] + count[i]

    # max = 255
    index = 255 / (h * w)

    # Image equalization
    for i in range(len(cdf)):
        x = int(round(index * cdf[i]))
        move[i] = x - i
```

```
    # Equalized image 생성
    outimg = np.zeros(shape = (h, w)).astype('uint8')
    for i in range(h):
        for j in range(w):
            outimg.itemset((i,j),(inimg[i,j] + move[inimg[i,j]]))

    plt.imshow(outimg,'gray',vmin=0,vmax=255)
    plt.show()
    plt.hist(outimg.flatten(),256,[0,256])
    plt.show()

    break
```

- Image



Input Image



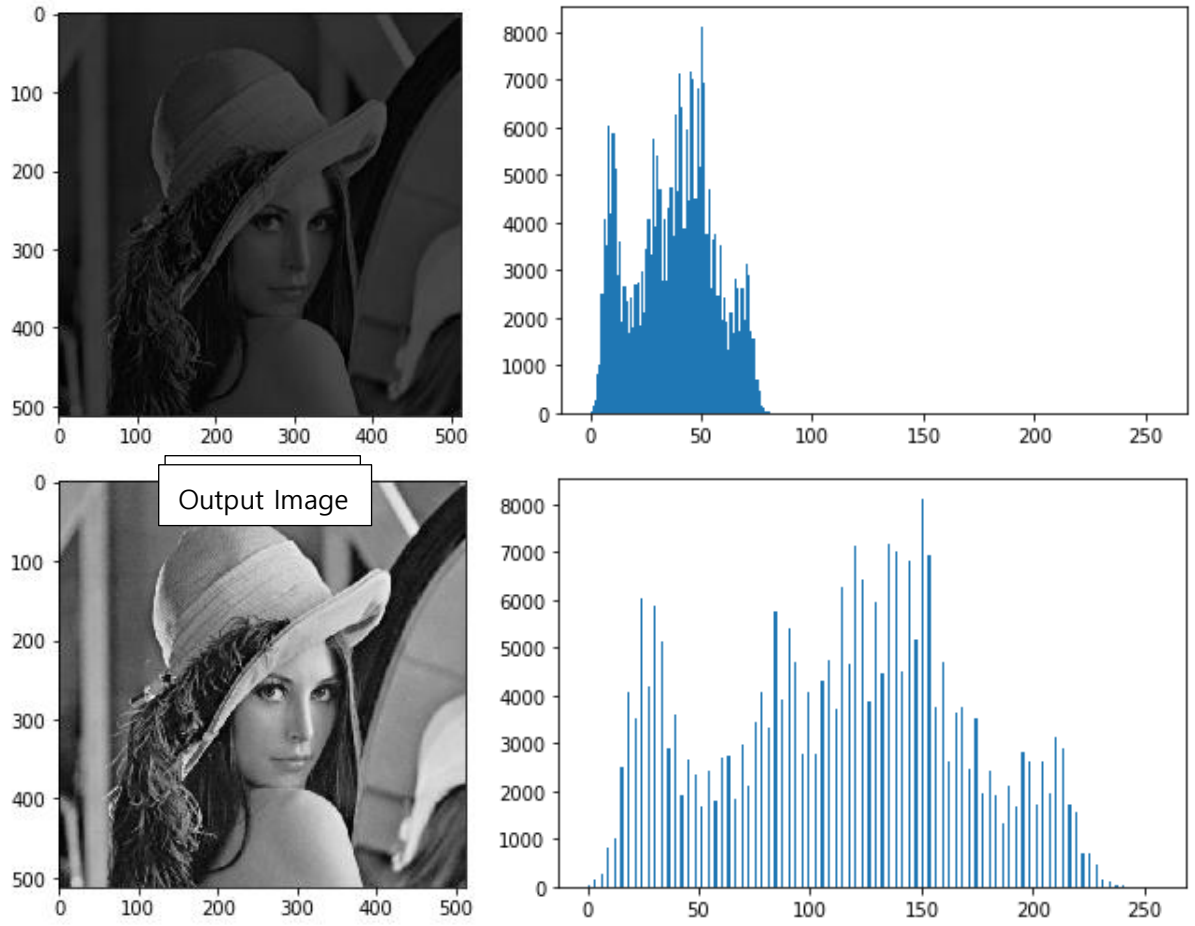
Output Image

contrast stretching

- Python Code

```
import scipy
import numpy as np
from scipy import ndimage
from matplotlib import pyplot as plt

inimg = plt.imread('lena_512_low1.png')
inimg = np.uint8(255*inimg)

plt.imshow(inimg,'gray',vmin=0,vmax=255)
plt.show()
plt.hist(inimg.flatten(),256,[0,256])
plt.show()

#Contrast stretching
h,w = inimg.shape
maxval = np.amax(inimg)
minval = np.amin(inimg)
print(maxval,minval)

outimg = np.zeros(inimg.shape)
outimg = 255*np.float32(inimg[:,:]-minval)/float(maxval-minval)
outimg = np.uint8(outimg)
plt.imshow(outimg,'gray',vmin=0,vmax=255)
plt.show()

plt.hist(outimg.flatten(),256,[0,256])
plt.show()
```

- Image

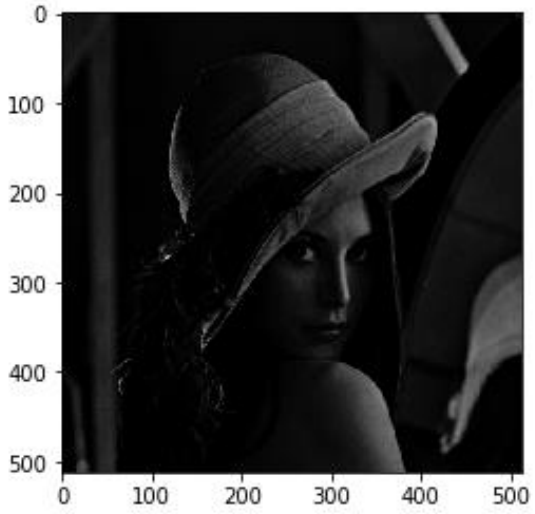


=> 그림1에 대한 histogram equalization과 contrast stretching한 결과를 각각 나타낸 것입니다. Contrast stretching은 단순히 영상 내 픽셀 밝기의 최솟값과 최댓값의 비율을 이용해서 고정된 비율을 가지고 영상을 낮은 밝기와 높은 밝기로 스트레칭 해주는 선형 정규화 기법인 반면, histogram equalization은 영상의 전체 픽셀값을 균등하게 재배치하는 비선형 정규화 기법입니다. 그러므로 histogram equalization는 밝기 값의 차이들이 적어서 contrast stretching보다 명암대비가 더 세밀하게 이루어집니다. 실제 결과 이미지를 비교해보면, 입력 이미지의 히스토그램이 0~100 사이에 치우쳐서 분포되었던 것이 contrast stretching과 histogram equalization해준 결과, 분포가 0~256 사이에 넓게 퍼진 것을 확인할 수 있습니다. 히스토그램을 더 면밀히 관찰해보면, 히스토그램이 펼쳐진 효과가 histogram equalization에서 더 크고, 더 넓고 균일하게 분포된 것을 확인할 수 있습니다. 또한, histogram equalization한 결과 영상이 contrast stretching한 결과 영상보다 훨씬 가시도가 좋고, 선명합니다.

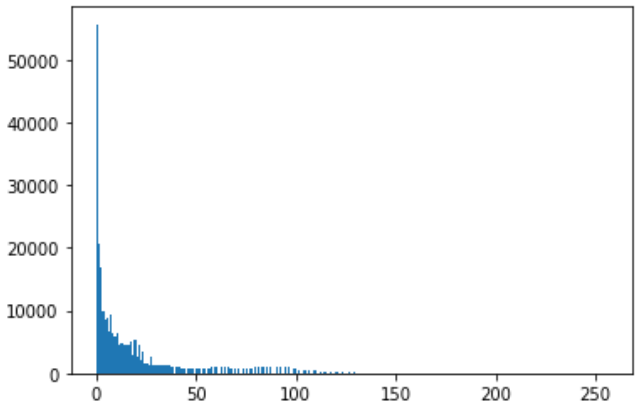
* 그림2 부터는 code는 그림1과 같으며, 이미지 결과만 비교하겠습니다.

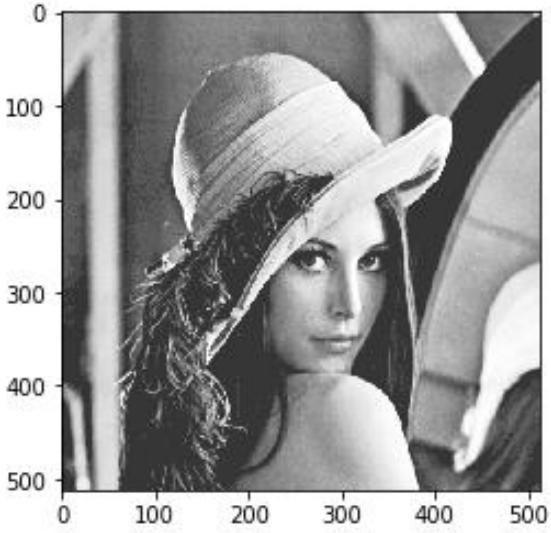
(2) 그림2 - lena_512_low2

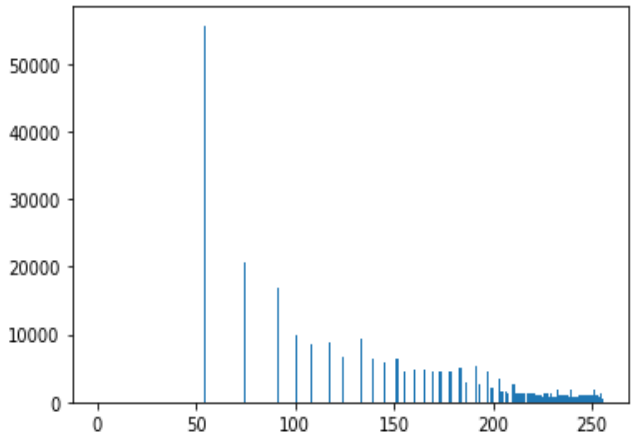
histogram equalization



Input Image

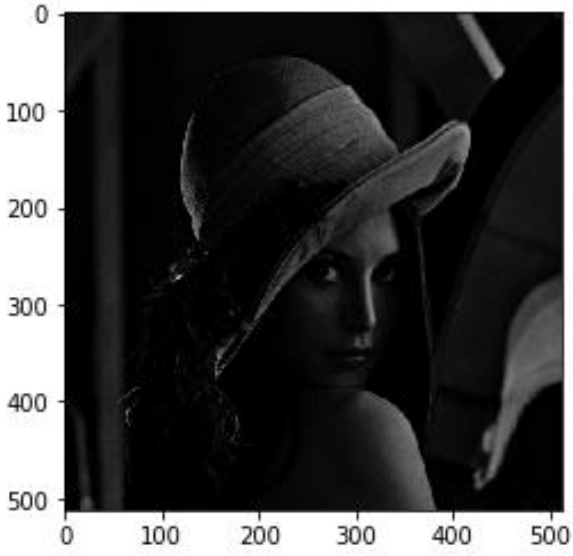


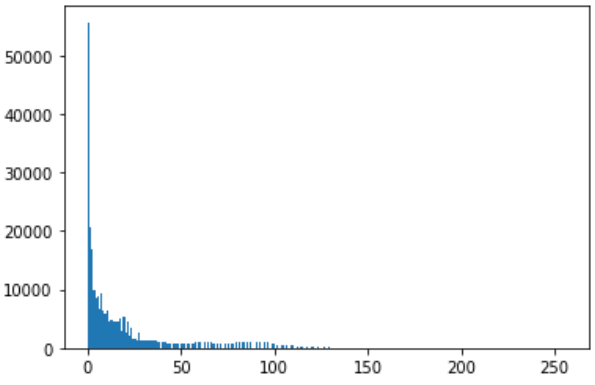




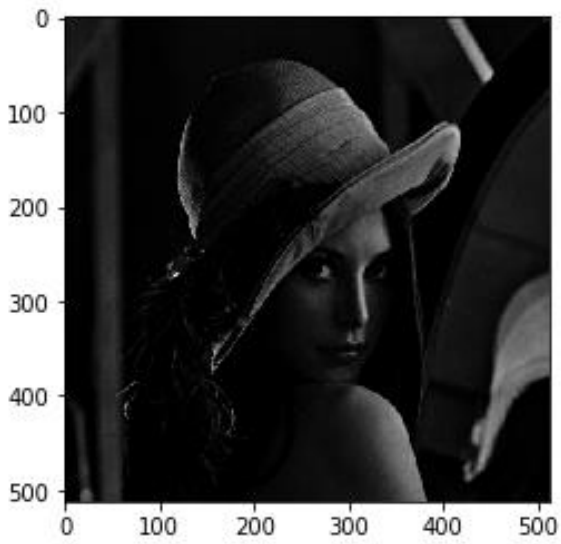
Output Image

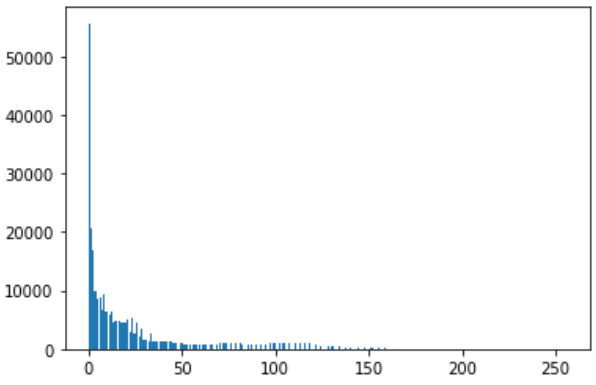
contrast stretching





Input Image



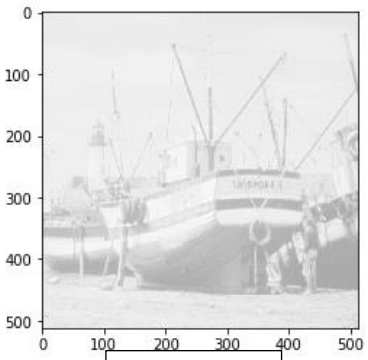


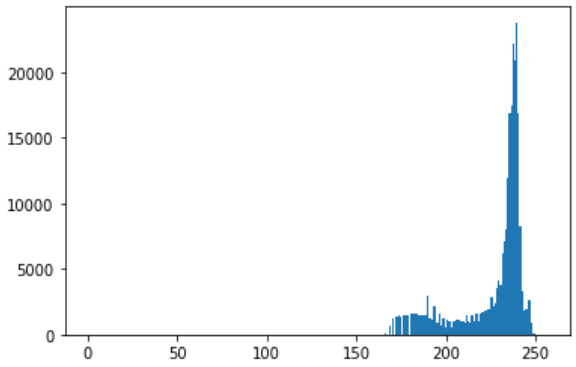
Output Image

=> 그림2에 대한 histogram equalization과 contrast stretching한 결과를 각각 나타낸 것입니다. 실제 결과 이미지를 비교해보면, histogram equalization 결과가 contrast stretching 결과 보다 히스토그램 분포가 더 퍼져있고, 출력 이미지도 명암대비가 훨씬 좋고 밝게 잘보입니다. 반면에 contrast stretching 결과는 히스토그램 분포가 입력 이미지와 비교했을 때 큰 차이가 없는 모습이고, 흐릿했던 기존 이미지보다는 화질은 향상되었습니다. 이렇게 명암대비가 거의 없는 영상은 histogram equalization기법을 적용하는 것이 더 적절해 보입니다.

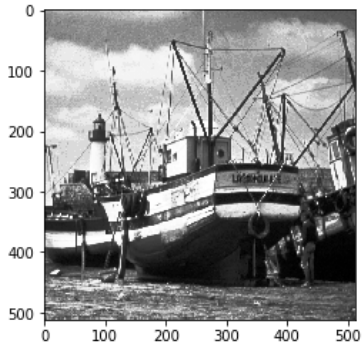
(3) 그림3 - boats_512_high.png

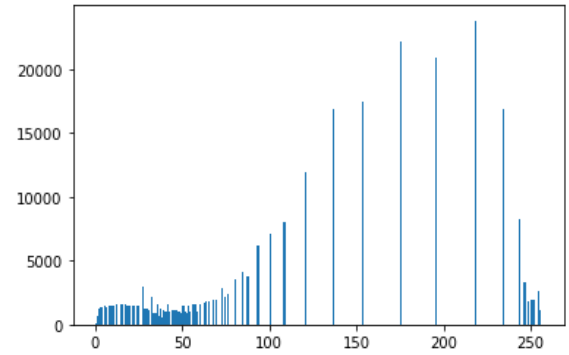
histogram equalization





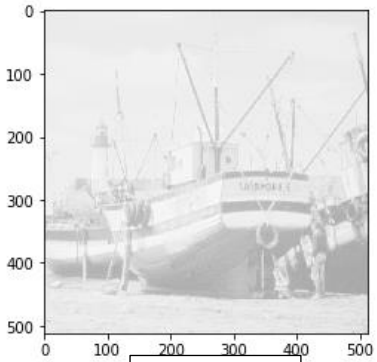
Input Image

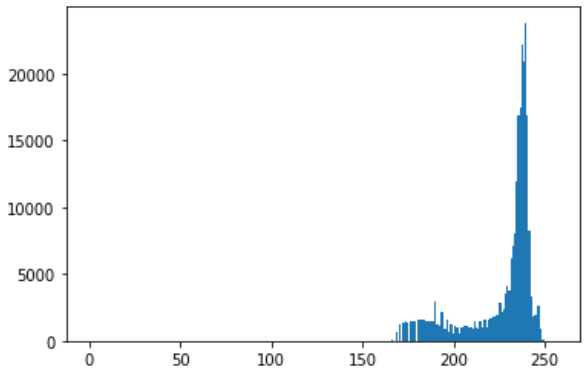




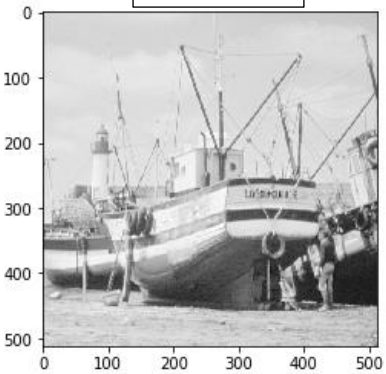
Output Image

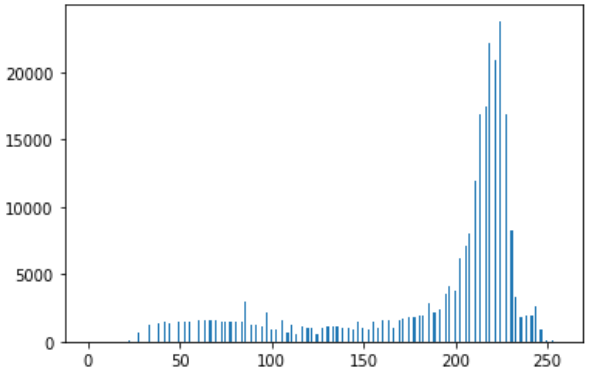
contrast stretching





Input Image



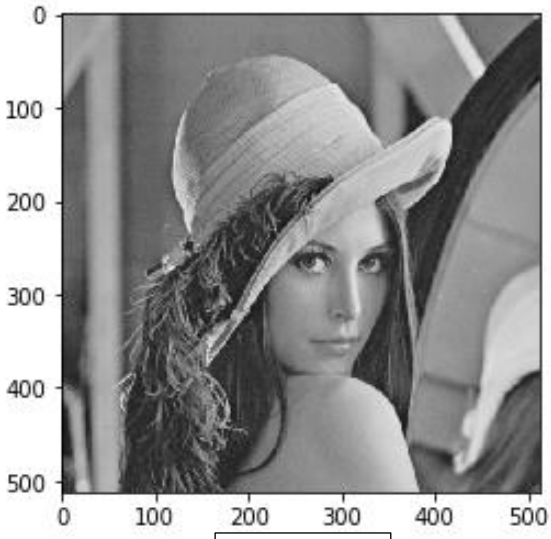


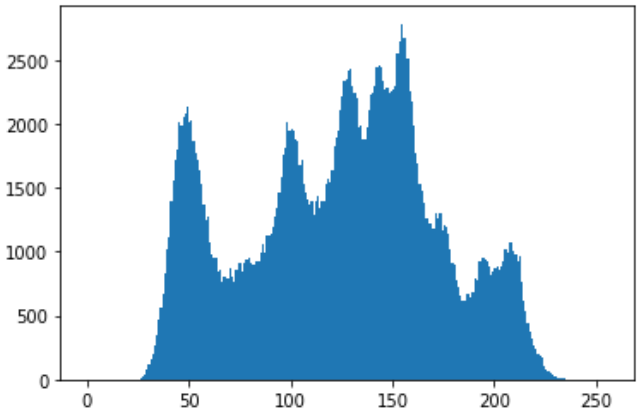
Output Image

=> 그림3에 대한 histogram equalization과 contrast stretching한 결과를 각각 나타낸 것입니다. 실제 결과 이미지를 비교해보면, histogram equalization 결과가 contrast stretching 결과 보다 히스토그램 분포가 더 넓고 균일하게 퍼져있고, 출력 이미지도 명암대비가 훨씬 좋고 선명하게 보입니다. 반면에 contrast stretching 결과는 히스토그램 분포가 입력 이미지와 비교했을 때 분포의 범위가 넓어지긴 했지만, 200~250 사이에 분포가 몰려있는 것은 달라지지 않았습니다.

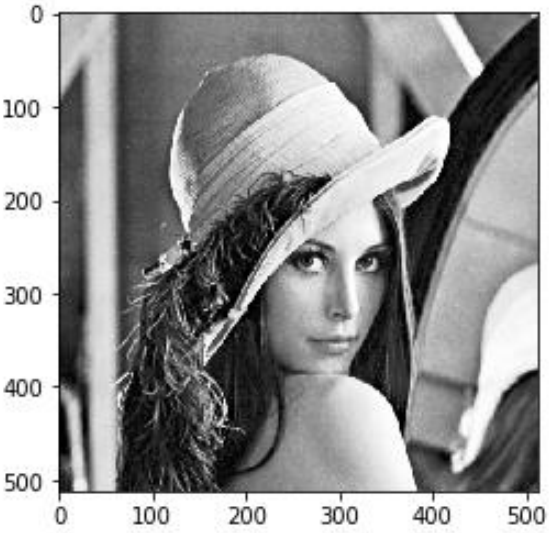
(4) 그림4 - lena.png

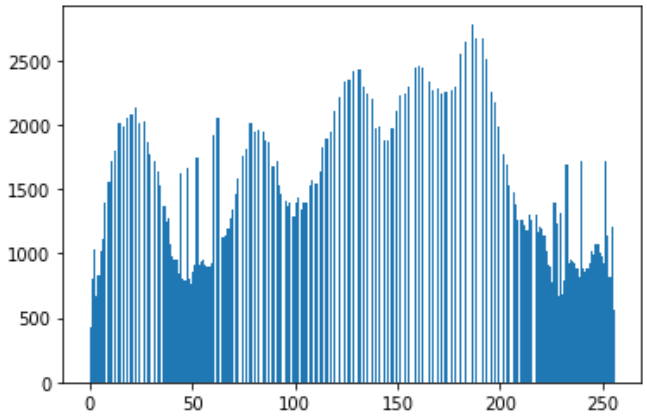
histogram equalization





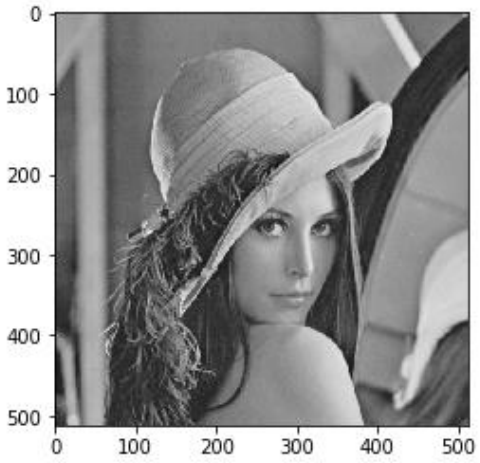
Input Image

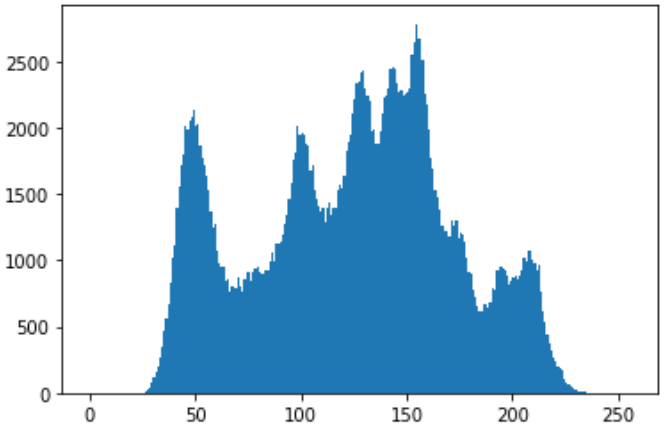




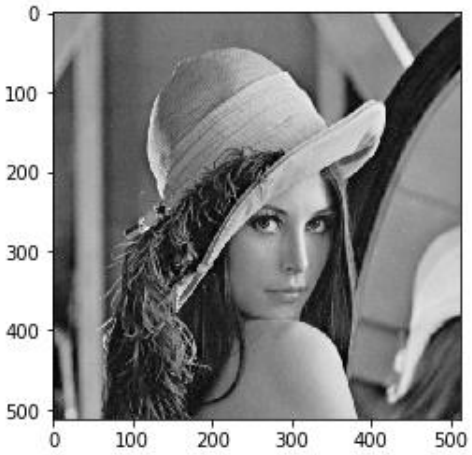
Output Image

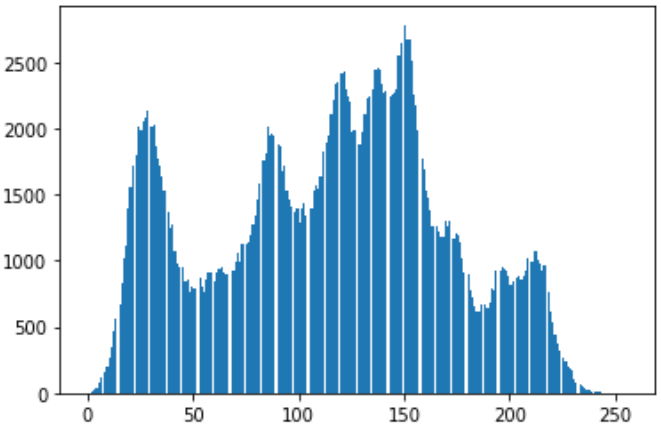
contrast stretching





Input Image





Output Image

=> 그림4에 대한 histogram equalization과 contrast stretching한 결과를 각각 나타낸 것입니다. 실제 결과 이미지를 비교해보면, histogram equalization 결과가 contrast stretching 결과 보다 히스토그램 분포가 더 넓고 균일하게 퍼져있고, 출력 이미지도 명암대비가 훨씬 좋고 선명하게 보입니다. 반면에, contrast stretching 결과는 히스토그램 분포가 입력 이미지와 비교했을 때 분포의 범위가 넓어지긴 했지만, 이미지는 약간의 화질 향상과 명암대비는 있지만, histogram equalization과 비교했을 때 상대적으로 크게 변화가 없습니다.

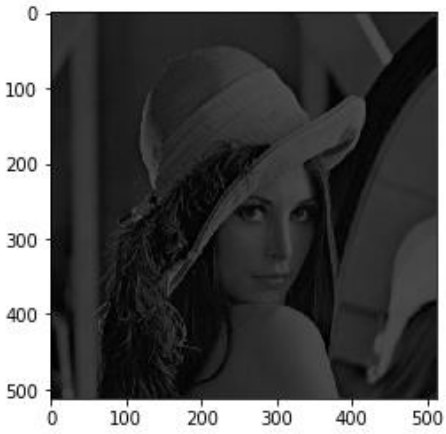
2. 1번에서 작성한 program에서 histogram equalization 부분을 function에서 수행하도록 code를 수정하고 동작을 검증하시오.

(1) 그림1 - lena_512_low1.png

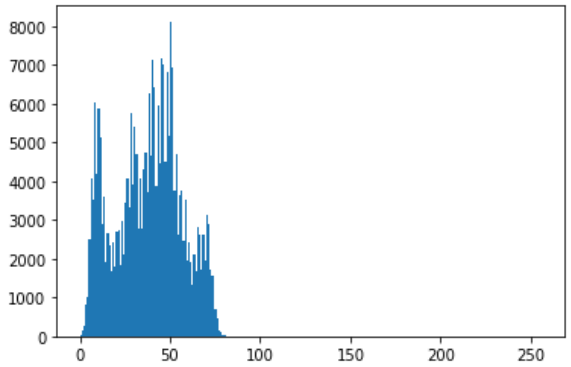
- Python Code

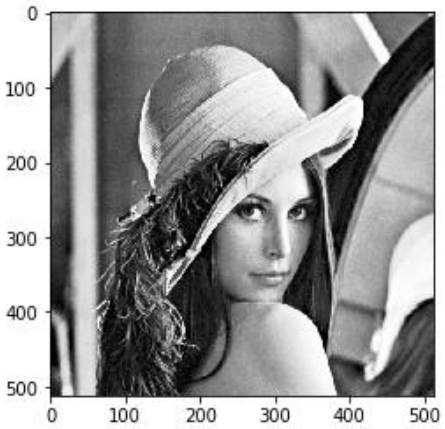
```
import numpy as np
from matplotlib import pyplot as plt

inimg = plt.imread('lena_512_low1.png')
inimg = np.uint8(255*inimg)
plt.imshow(inimg,'gray',vmin=0,vmax=255)
plt.show()
plt.hist(inimg.flatten(),256,[0,256])
plt.show()

hist, bin = np.histogram(inimg.flatten(), 256, [0,256])

# 함수 적용
cdf = hist.cumsum()
cdf_m = np.ma.masked_equal(cdf,0)

print(cdf_m.min())
print(cdf_m.max())
cdf_m = (cdf_m)/(cdf_m.max())*(256-1)
cdf = np.ma.filled(cdf_m,0).astype('uint8')
outimg = cdf[inimg]

plt.imshow(outimg,'gray',vmin=0,vmax=255)
plt.show()
plt.hist(outimg.flatten(),256,[0,256])
plt.show()
```

- Image

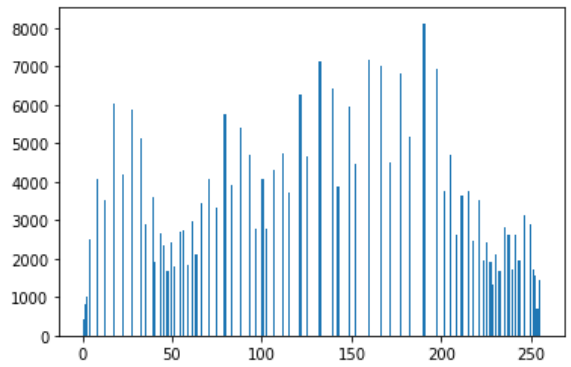


Input Image





Output Image

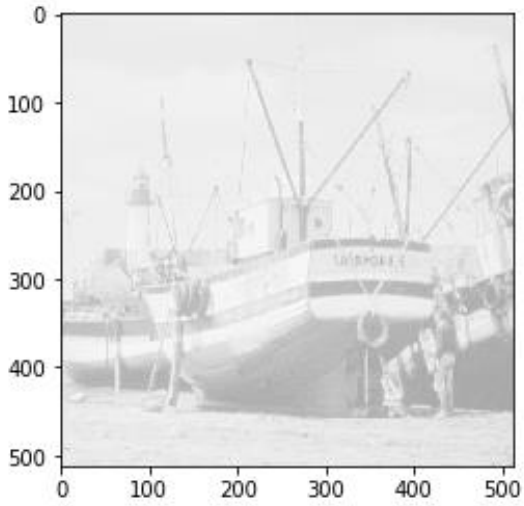


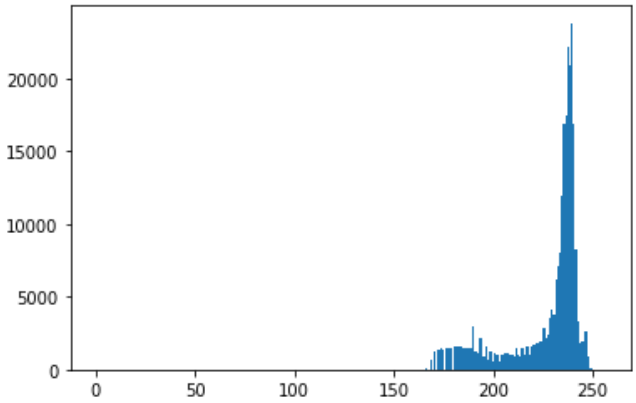
=> Histogram equalization을 function을 이용하여 실행시킨 결과, 1-(1)과 결과가 같음을 확인하였습니다.

* 그림2의 code는 그림1과 같으며, 이미지 결과만 비교하겠습니다.

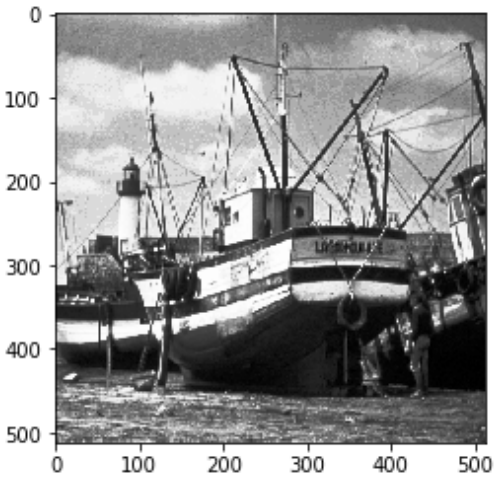
(2) 그림2 - boats_512_high.png

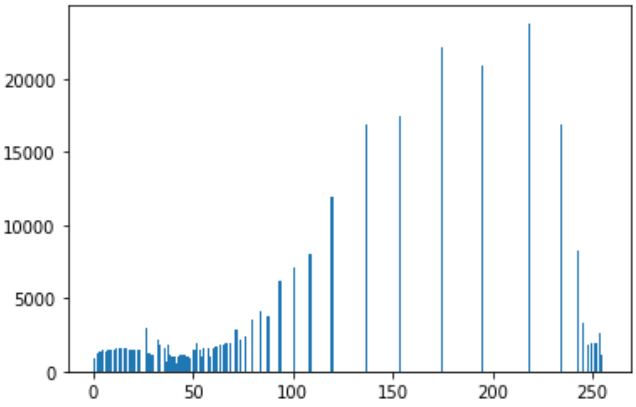
- Image





Input Image





Output Image

=> Histogram equalization을 function을 이용하여 실행시킨 결과, 1-(3)과 결과가 같음을 확인하였습니다.

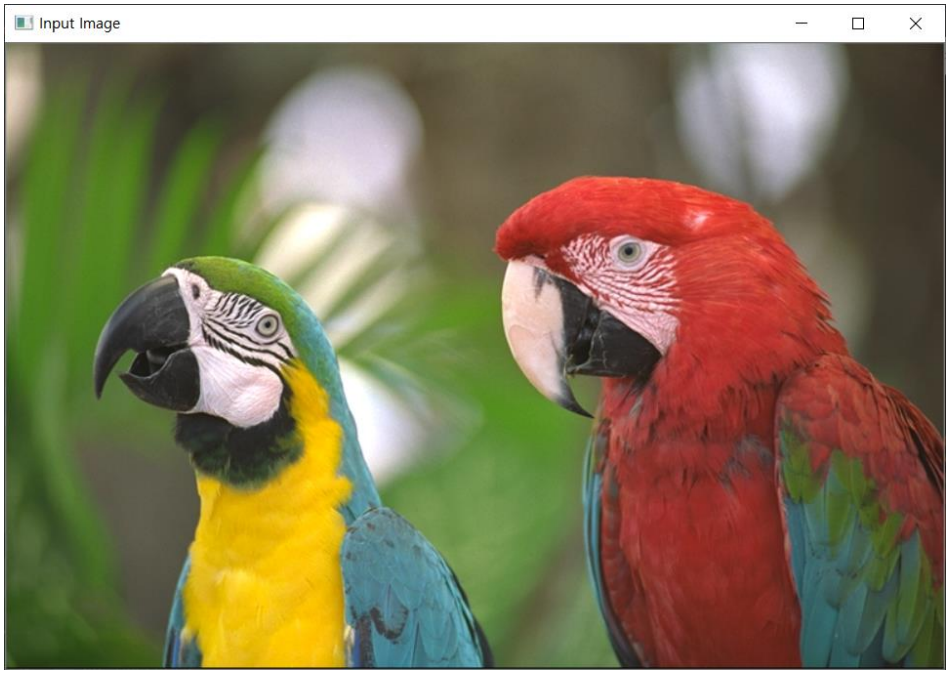
3. 1번에서 작성한 program을 color 영상에 대해서 수행하도록 다시 작성하고 두 개의 color 영상에 대해서 실행한 후에 결과를 논하시오. (RGB 각각 histogram equalization)

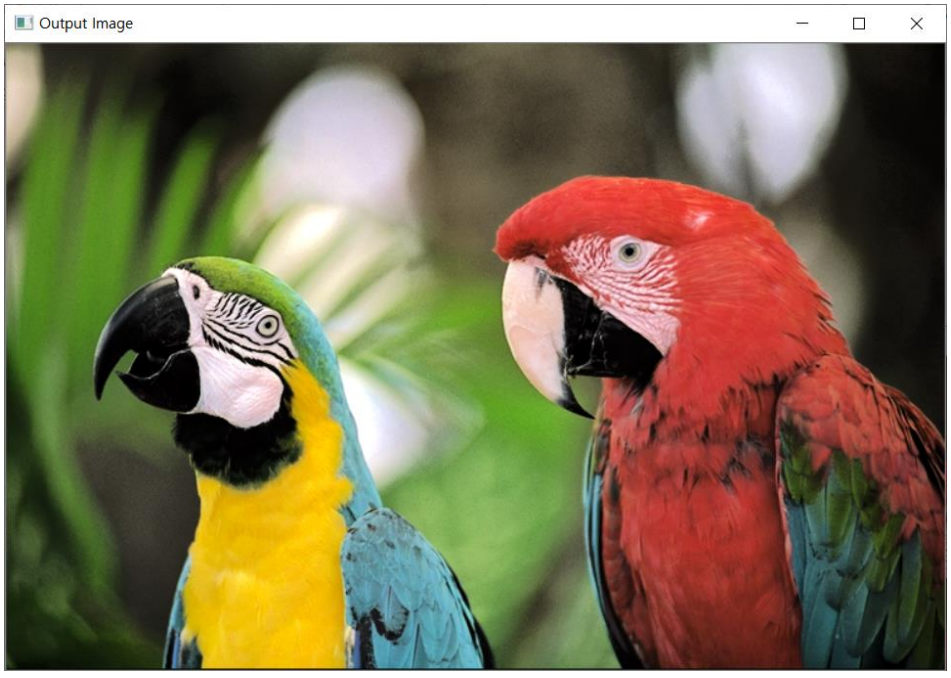
(1) 그림1 - parrot.png

- Python Code

```
import cv2


# 영상을 BGR 스케일로 읽기
inimg = cv2.imread("parrot.png")

# 컬러 스케일을 BGR에서 HSV로 변경
img_hsv = cv2.cvtColor(inimg, cv2.COLOR_BGR2HSV)

# HSV 컬러 스케일의 마지막 채널(명도)에 대해 histogram equalization 적용
img_hsv[:, :, 2] = cv2.equalizeHist(img_hsv[:, :, 2])

# 컬러 스케일을 HSV에서 BGR로 변경
outimg = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)

cv2.imshow('Input Image', inimg)
cv2.imshow('Output Image', outimg)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Image



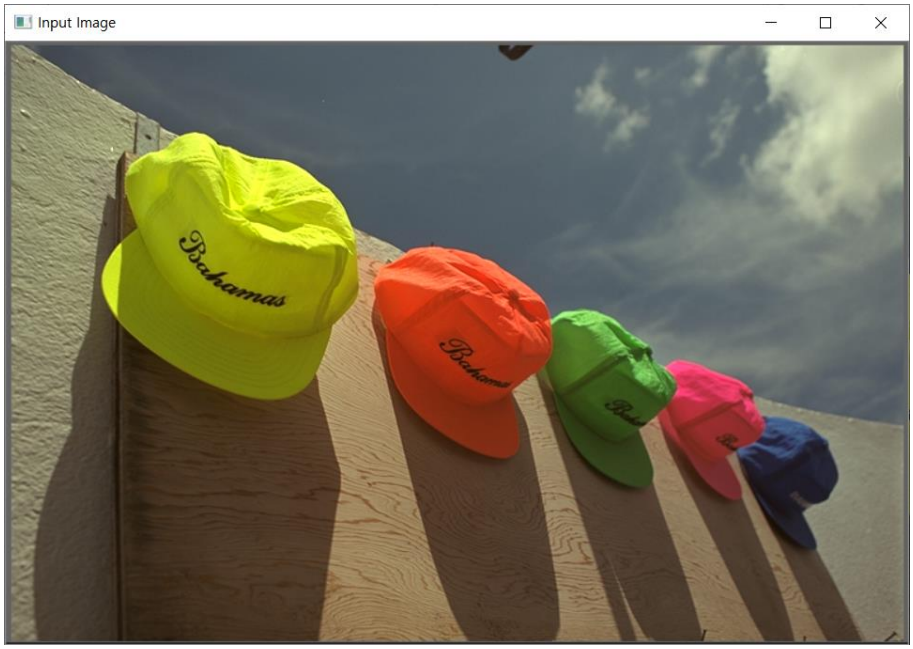


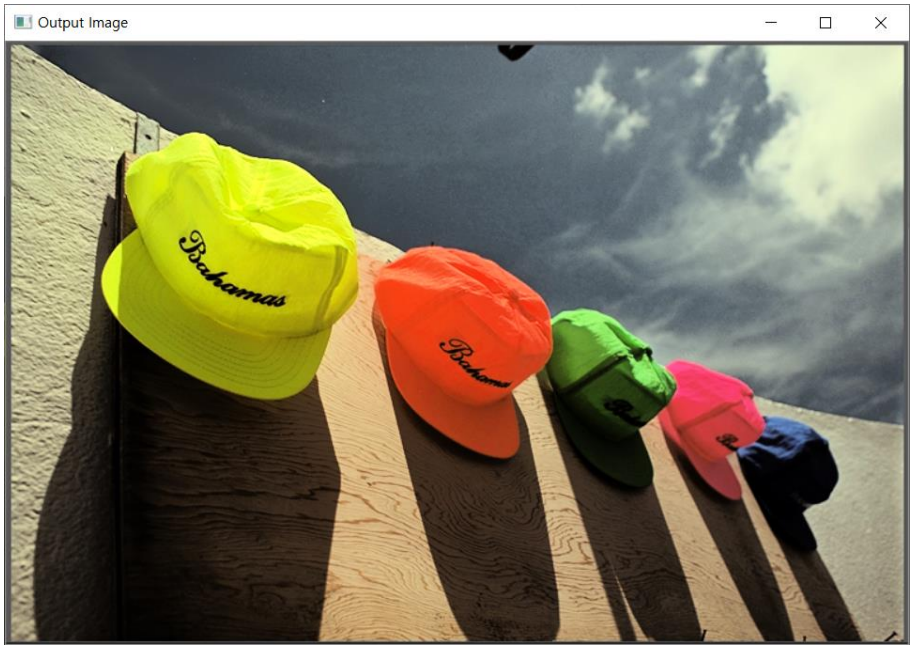
(1) 그림2 - cap.png

- Python Code

```
import cv2


# 영상을 BGR 스케일로 읽기
inimg = cv2.imread("cap.png")

# 컬러 스케일을 BGR에서 HSV로 변경
img_hsv = cv2.cvtColor(inimg, cv2.COLOR_BGR2HSV)

# HSV 컬러 스케일의 마지막 채널(명도)에 대해 histogram equalization 적용
img_hsv[:, :, 2] = cv2.equalizeHist(img_hsv[:, :, 2])

# 컬러 스케일을 HSV에서 BGR로 변경
outimg = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)

cv2.imshow('Input Image', inimg)
cv2.imshow('Output Image', outimg)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Image





=> 그림1과 그림2에 histogram equalization을 적용하여 각각의 결과를 나타낸 것입니다. 두 이미지 모두 결과이미지의 명암대비가 훨씬 좋고 밝게 잘보입니다.